

Reproducible computational environment, when?

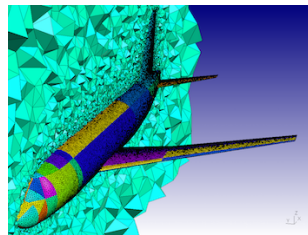
*How to redone later and over-there
what had be done today and here?*

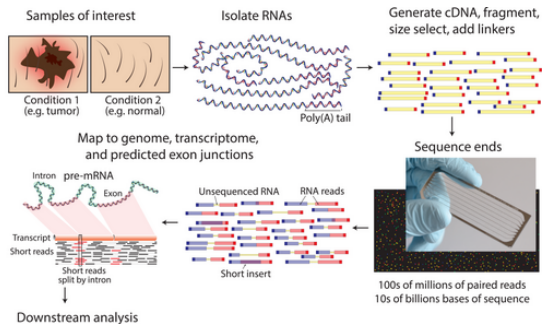
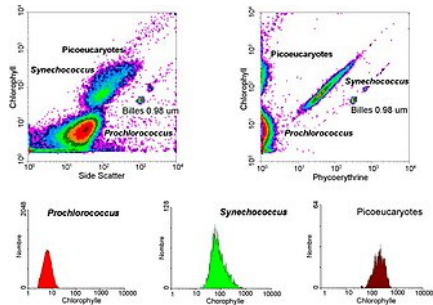
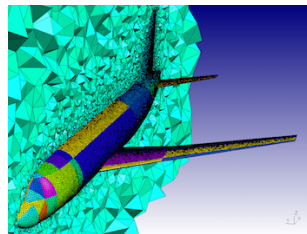
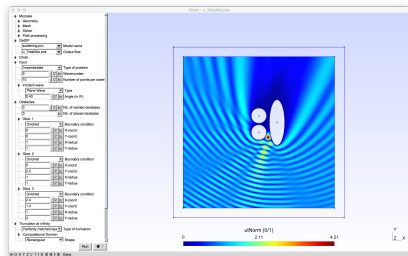
Simon Tournier

UMR 1342 - IRSL
`simon.tournier@inserm.fr`
`@zimoun@sciences.re`

November 12th, 2025







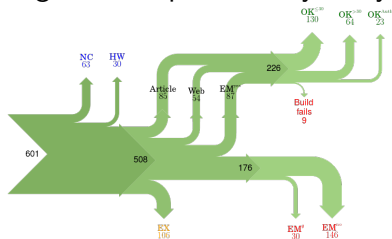
- ▶ ...-2016: Student, PhD, Postdoc. My Happy Life! Science rocks!

- ▶ ...-2016: Student, PhD, Postdoc. My Happy Life! Science rocks!
- ▶ 2016?: Attend Roberto Di Cosmo's talk (organized by EDF)

- ▶ ...-2016: Student, PhD, Postdoc. My Happy Life! Science rocks!
- ▶ 2016?: Attend Roberto Di Cosmo's talk (organized by EDF)

Bang!

Collberg's 2015 reproducibility study ([link](#))



601 mainstream papers

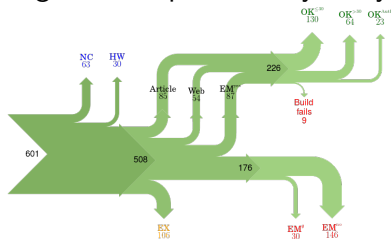
- ▶ 508 with tools
- ▶ **only 40%** *installable*

(even computer science is far from ideal!)

- ▶ ...-2016: Student, PhD, Postdoc. My Happy Life! Science rocks!
- ▶ 2016?: Attend Roberto Di Cosmo's talk (organized by EDF)

Bang!

Collberg's 2015 reproducibility study ([link](#))



601 mainstream papers

- ▶ 508 with tools
- ▶ **only 40%** *installable*

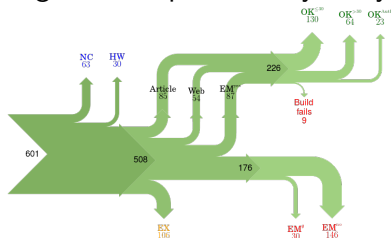
(even computer science is far from ideal!)

- ▶ 2016-...: Start a permanent position in *biomedical* environment

- ▶ ...-2016: Student, PhD, Postdoc. My Happy Life! Science rocks!
- ▶ 2016?: Attend Roberto Di Cosmo's talk (organized by EDF)

Bang!

Collberg's 2015 reproducibility study ([link](#))



601 mainstream papers

- ▶ 508 with tools
- ▶ **only 40%** *installable*

(even computer science is far from ideal!)

- ▶ 2016-...: Start a permanent position in *biomedical* environment
 - ▶ Producing *results* implies many tools.
 - ▶ Spending too many resource fighting against these tools.
 - ▶ Science rocks. . . less!

Ok, that's an issue!
why? what can we do?

Replication and reproducibility crisis

More than 70% of researchers have tried and **failed to reproduce** another scientist's experiments, and more than half have failed to reproduce their own experiments.

1,500 scientists lift the lid on reproducibility (Nature, 2016) [\(link\)](#)

Replication and reproducibility crisis

More than 70% of researchers have tried and **failed to reproduce** another scientist's experiments, and more than half have failed to reproduce their own experiments.

1,500 scientists lift the lid on reproducibility (Nature, 2016) ([link](#))

- ▶ « ReScience challenge »: try to redo ten years old paper.
- ▶ Reprohackathons: promoting reproducibility in bioinformatics through training ([link](#))

Replication and reproducibility crisis

More than 70% of researchers have tried and **failed to reproduce** another scientist's experiments, and more than half have failed to reproduce their own experiments.

1,500 scientists lift the lid on reproducibility (Nature, 2016) ([link](#))

- ▶ « ReScience challenge »: try to redo ten years old paper.
- ▶ Reprohackathons: promoting reproducibility in bioinformatics through training ([link](#))

Many causes... one solution?
at least, *Open Science* helps

(reproductibility	=	verification)
	replicability	=	validation	
	redo			

Open Science Reproducible Research

Science = Transparent and Collective
Scientific result = Experiment + Numerical treatment

Science at the digital age:

- | | |
|-----------------|---------------------------|
| 1. Open Article | HAL, BioArxiv |
| 2. Open Data | Data Repositories, Zenodo |
| 3. Open Source | Forges, GitLab |

~~Open~~ Science Reproducible Research

Science = Transparent and Collective
Scientific result = Experiment + *Numerical treatment*

Science at the digital age:

- | | |
|-----------------|---------------------------|
| 1. Open Article | HAL, BioArxiv |
| 2. Open Data | Data Repositories, Zenodo |
| 3. Open Source | Forges, GitLab |

how to *glue* all that?

~~Open~~ Science Reproducible Research

Science = Transparent and Collective
Scientific result = Experiment + *Numerical treatment*

Science at the digital age:

- | | |
|------------------------------|--|
| 1. Open Article | HAL, BioArxiv |
| 2. Open Data | Data Repositories, Zenodo |
| 3. Open Source | Forges, GitLab, <i>Software Heritage</i> |
| 4. <i>Computational env.</i> | ? |

how to *glue* all that?

Open-Science Reproducible Research

Science = Transparent and Collective
Scientific result = Experiment + *Numerical treatment*

Science at the digital age:

- | | | |
|----|---------------------------|--|
| 1. | Open Article | HAL, BioArxiv |
| 2. | Open Data | Data Repositories, Zenodo |
| 3. | Open Source | Forges, GitLab, <i>Software Heritage</i> |
| 4. | <i>Computational env.</i> | ? |

how to *glue* all that?

today's topic
considering long-term (1-5 years)

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
result	←	paper	+	data	+	analysis
data	←	protocol	+	instruments	+	materials
analysis	←	script	+	data	+	environment

- ▶ **audit** is the « tractable » part
- ▶ **opaque** is generally the hard part

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
result	←	paper	+	data	+	analysis
data	←	protocol	+	instruments	+	materials
analysis	←	script	+	data	+	environment

- ▶ **audit** is the « tractable » part
- ▶ **opaque** is generally the hard part
- ▶ how to evacuate **depend?** from the equations

Redo (reproduce or replicate) a result?

audit

opaque

depend?

result	←	paper	+	data	+	analysis
data	←	protocol	+	instruments	+	materials
★ analysis	←	script	+	data	+	environment

- ▶ **audit** is the « tractable » part
- ▶ **opaque** is generally the hard part
- ▶ how to evacuate **depend?** from the equations

★ *our issue*

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
result	←	paper	+	data	+	analysis
data	←	protocol	+	instruments	+	materials
★ analysis	←	script	+	data	+	environment

- ▶ **audit** is the « tractable » part
- ▶ **opaque** is generally the hard part
- ▶ how to evacuate **depend?** from the equations...
...try to turn **environment** into **audit**

★ *our issue*

Redo (reproduce or replicate) a result?

		audit		opaque		depend?
result	←	paper	+	data	+	analysis
data	←	protocol	+	instruments	+	materials
★ analysis	←	script	+	data	+	environment

- ▶ **audit** is the « tractable » part
- ▶ **opaque** is generally the hard part
- ▶ how to evacuate **depend?** from the equations...
...try to turn **environment** into **audit**

★ *our issue* $\left(\begin{array}{ccc} \text{« computer »} \approx \text{instrument} & \text{and} & \text{« computation »} \approx \text{measurement} \\ \text{computational env.} & \leftrightarrow & \text{experimental setup} \end{array} \right)$

Challenges about Reproducible Research

From the « scientific method » viewpoint:

controlling the source of variations

⇒ transparent

as with instrument \approx computer

From the « scientific knowledge » viewpoint:

(universal?)

- ▶ Independant observer must be able to observe the same result.
- ▶ The observation must be sustainable (to some extent).

⇒ collective

Challenges about Reproducible Research

From the « scientific method » viewpoint:

controlling the source of variations

⇒ transparent

as with instrument \approx computer

From the « scientific knowledge » viewpoint:

(universal?)

- ▶ Independant observer must be able to observe the same result.
- ▶ The observation must be sustainable (to some extent).

⇒ collective

In a world where (almost) all is *data*

how to redo later and elsewhere what has been done today and here?

(implicitly using a « computer »)

Agence Nationale de Sécurité du Médicament et des produits de santé (ANSM)

- ① Logiciels et applications mobiles en santé (lien)

*Certains de ces **logiciels** sont des **dispositifs médicaux** (DM) ou des dispositifs médicaux de diagnostic in vitro (DM DIV), car ils ont une finalité médicale.*

Agence Nationale de Sécurité du Médicament et des produits de santé (ANSM)

- 1 Logiciels et applications mobiles en santé ([lien](#))

*Certains de ces **logiciels** sont des **dispositifs médicaux** (DM) ou des dispositifs médicaux de diagnostic in vitro (DM DIV), car ils ont une finalité médicale.*

- 2 Traçabilité des dispositifs médicaux ([link](#))

***l'identification unique** des dispositifs et afficher au minimum:*

Agence Nationale de Sécurité du Médicament et des produits de santé (ANSM)

① Logiciels et applications mobiles en santé ([lien](#))

*Certains de ces **logiciels** sont des **dispositifs médicaux** (DM) ou des dispositifs médicaux de diagnostic in vitro (DM DIV), car ils ont une finalité médicale.*

② Traçabilité des dispositifs médicaux ([link](#))

*l'**identification unique** des dispositifs et afficher au minimum:*

- ▶ *la dénomination ou la référence du produit,*
- ▶ *le nom ou la référence du fabricant ou de son mandataire,*
- ▶ *le numéro de lot ou de série du produit.*

Agence Nationale de Sécurité du Médicament et des produits de santé (ANSM)

① Logiciels et applications mobiles en santé ([lien](#))

*Certains de ces **logiciels** sont des **dispositifs médicaux** (DM) ou des dispositifs médicaux de diagnostic in vitro (DM DIV), car ils ont une finalité médicale.*

② Traçabilité des dispositifs médicaux ([link](#))

*l'**identification unique** des dispositifs et afficher au minimum:*

- ▶ *la dénomination ou la référence du produit,*
- ▶ *le nom ou la référence du fabricant ou de son mandataire,*
- ▶ *le numéro de lot ou de série du produit.*

Agence Nationale de Sécurité du Médicament et des produits de santé (ANSM)

① Logiciels et applications mobiles en santé (lien)

*Certains de ces **logiciels** sont des **dispositifs médicaux** (DM) ou des dispositifs médicaux de diagnostic in vitro (DM DIV), car ils ont une finalité médicale.*

② Traçabilité des dispositifs médicaux (link)

***l'identification unique** des dispositifs et afficher au minimum:*

- ▶ *la dénomination ou la référence du produit,*
- ▶ *le nom ou la référence du fabricant ou de son mandataire,*
- ▶ *le numéro de lot ou de série du produit.*

Software Bill of Materials (SBOM) and supply chain

Guix blog post: Identifying software (link)

Building Medical Devices with Debian (DebConf19) (lien)

Agence Nationale de Sécurité du Médicament et des produits de santé (ANSM)

① Logiciels et applications mobiles en santé (lien)

*Certains de ces **logiciels** sont des **dispositifs médicaux** (DM) ou des dispositifs médicaux de diagnostic in vitro (DM DIV), car ils ont une finalité médicale.*

② Traçabilité des dispositifs médicaux (link)

*l'**identification unique** des dispositifs et afficher au minimum:*

- ▶ *la dénomination ou la référence du produit,*
- ▶ *le nom ou la référence du fabricant ou de son mandataire,*
- ▶ *le numéro de lot ou de série du produit.*

software = medical device \implies unique identification
traceability?

Software Bill of Materials (SBOM) and supply chain

Guix blog post: Identifying software (link)

Building Medical Devices with Debian (DebConf19) (lien)

Archive

Research software artifacts must be properly **archived**
make sure we can **retrieve** them (*reproducibility*)

Reference

Research software artifacts must be properly **referenced**
make sure we can **identify** them (*reproducibility*)

Describe

Research software artifacts must be properly **described**
make it easy to *discover* and **reuse** them (*visibility*)

Cite/Credit

Research software artifacts must be properly **cited** (*not the same as referenced!*)
to give **credit** to authors (*evaluation!*)

We will speak about...

- 1 Situation
- 2 The problem of Alice and Blake
 - Capturing what?
 - How to capture?
- 3 The Guix's way
- 4 Source code archiving
 - Software Heritage
- 5 Bridging?

(some examples from C programming language but all apply equally to any other computational stack)
Python, R, Julia, etc.

Capturing what?

Software is dual

human-readable vs machine-readable

source + transformation → binary

Software is dual

human-readable vs machine-readable

source + transformation → binary

Program (source code)

```
/* Hello World program */  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```


Software is dual

human-readable vs machine-readable

source + transformation → binary

Program (source code)

```
/* Hello World program */  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```

Software is dual

human-readable vs machine-readable

source + transformation → binary

Program (source code)

```
/* Hello World program */  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```



Program (excerpt of binary)

```
4004e6: 55  
4004e7: 48 89 e5  
4004ea: bf 84 05 40 00  
4004ef: b8 00 00 00 00  
4004f4: e8 c7 fe ff ff  
4004f9: 90  
4004fa: 5d  
4004fb: c3
```

Software is dual

human-readable vs machine-readable

source + transformation → binary

Program (source code)

```
/* Hello World program */  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```



Program (excerpt of binary)

```
4004e6: 55  
4004e7: 48 89 e5  
4004ea: bf 84 05 40 00  
4004ef: b8 00 00 00 00  
4004f4: e8 c7 fe ff ff  
4004f9: 90  
4004fa: 5d  
4004fb: c3
```

transparency?

Software is dual

human-readable vs machine-readable

source + transformation → binary

Program (source code)

```
/* Hello World program */  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```



Program (excerpt of binary)

```
4004e6: 55  
4004e7: 48 89 e5  
4004ea: bf 84 05 40 00  
4004ef: b8 00 00 00 00  
4004f4: e8 c7 fe ff ff  
4004f9: 90  
4004fa: 5d  
4004fb: c3
```

transparency

Software is dual

human-readable vs machine-readable

source + transformation → binary

Program (source code)

```
/* Hello World program */  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```



Program (excerpt of binary)

```
4004e6: 55  
4004e7: 48 89 e5  
4004ea: bf 84 05 40 00  
4004ef: b8 00 00 00 00  
4004f4: e8 c7 fe ff ff  
4004f9: 90  
4004fa: 5d  
4004fb: c3
```

long-term? transparency

Software is dual

human-readable vs machine-readable

source + transformation → binary

Program (source code)

```
/* Hello World program */  
  
#include<stdio.h>  
  
void main()  
{  
    printf("Hello World");  
}
```



Program (excerpt of binary)

```
4004e6: 55  
4004e7: 48 89 e5  
4004ea: bf 84 05 40 00  
4004ef: b8 00 00 00 00  
4004f4: e8 c7 fe ff ff  
4004f9: 90  
4004fa: 5d  
4004fb: c3
```

Questions (1/2)

Bessel function J_0 using C programming language

```
#include <stdio.h>
#include <math.h>

int main(){
    printf("%E\n", j0f(0x1.33d152p+1f));
}
```

Questions (1/2)

Bessel function J_0 using C programming language

```
#include <stdio.h>
#include <math.h>

int main(){
    printf("%E\n", j0f(0x1.33d152p+1f));
}
```

Alice sees: 5.643440E-08
Blake sees: 5.963430E-08

Determine if the difference is significant or not is let to experts, scientific field by scientific field

Questions (1/2)

Bessel function J_0 using C programming language

```
#include <stdio.h>
#include <math.h>

int main(){
    printf( "%E\n", j0f(0x1.33d152p+1f));
}
```

Alice sees: 5.643440E-08
Blake sees: 5.963430E-08

Why? In spite of everything being available (*open*)

Determine if the difference is significant or not is let to experts, scientific field by scientific field

Capturing what?

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
alice@laptop$
```

```
5.643440E-08
```

```
blake@desktop$
```

```
5.963430E-08
```

** Not an issue with floating-point computations*

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
alice@laptop$ gcc bess1.c                && ./a.out
5.643440E-08
blake@desktop$ gcc bess1.c -lm -fno-builtin && ./a.out
5.963430E-08
```

(due to *constant folding***)

* *Not an issue with floating-point computations*

** *C language is an example, other source but similar issues with Python, R, Perl, etc.*

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
alice@laptop$ gcc bess1.c                && ./a.out
5.643440E-08
blake@desktop$ gcc bess1.c -lm -fno-builtin && ./a.out
5.963430E-08
```

(due to *constant folding***)

Alice and Blake are running **two different computationnal environments**

* *Not an issue with floating-point computations*

** *C language is an example, other source but similar issues with Python, R, Perl, etc.*

Questions (2/2)

Alice and Blake both run « GCC at version 11.2.0 »

still different*

```
alice@laptop$ gcc bess1.c                && ./a.out
5.643440E-08
blake@desktop$ gcc bess1.c -lm -fno-builtin && ./a.out
5.963430E-08
```

(due to *constant folding***)

Alice and Blake are running **two different computationnal environments**

More than version number is required

* *Not an issue with floating-point computations*

** *C language is an example, other source but similar issues with Python, R, Perl, etc.*

How to capture?

Questions about a computational environment

- ▶ What is the code source?
- ▶ What are the tools required for building?
- ▶ What are the tools required for running?
- ▶ And recursively for each tool. . .

How to capture?

Questions about a computational environment

- ▶ What is the code source?
- ▶ What are the tools required for building?
- ▶ What are the tools required for running?
- ▶ And recursively for each tool. . .

Answering these questions enables **control over sources of variations**

Questions about a computational environment

- ▶ What is the code source?
- ▶ What are the tools required for building?
- ▶ What are the tools required for running?
- ▶ And recursively for each tool. . .

Answering these questions enables **control over sources of variations**

How to capture the answer of these questions?

*Usually: package manager (Conda, APT, Brew, . . .); Modulefiles; container; etc. ⇒ **not enough!***

How to capture?

Questions about a computational environment

- ▶ What is the code source?
- ▶ What are the tools required for building?
- ▶ What are the tools required for running?
- ▶ And recursively for each tool. . .

Answering these questions enables **control over sources of variations**

How to capture the answer of these questions?

*Usually: package manager (Conda, APT, Brew, . . .); Modulefiles; container; etc. ⇒ **not enough!***

toward a solution: Guix

How to capture?

Usual solutions and their difficulties

- 1 package manager: Conda, pip, `install.packages()`, APT (Debian/Ubuntu), etc.
- 2 container : Docker, Singularity

Usual solutions and their difficulties

- ① package manager: Conda, pip, `install.packages()`, APT (Debian/Ubuntu), etc.
- ② container : Docker, Singularity

① How to reinstall this exact same set of packages?

`apt-get update`

② How to inspect the base image?

`FROM debian:stable`

Usual solutions and their difficulties

- 1 package manager: Conda, pip, `install.packages()`, APT (Debian/Ubuntu), etc.
- 2 container : Docker, Singularity

1 How to reinstall this exact same set of packages?

`apt-get update`

2 How to inspect the base image?

`FROM debian:stable`

What if answering one question answers the other?

How to capture?

Usual solutions and their difficulties

- ① package manager: Conda, pip, `install.packages()`, APT (Debian/Ubuntu), etc.
- ② container : Docker, Singularity

$$\text{Guix} = \#1 + \#2$$

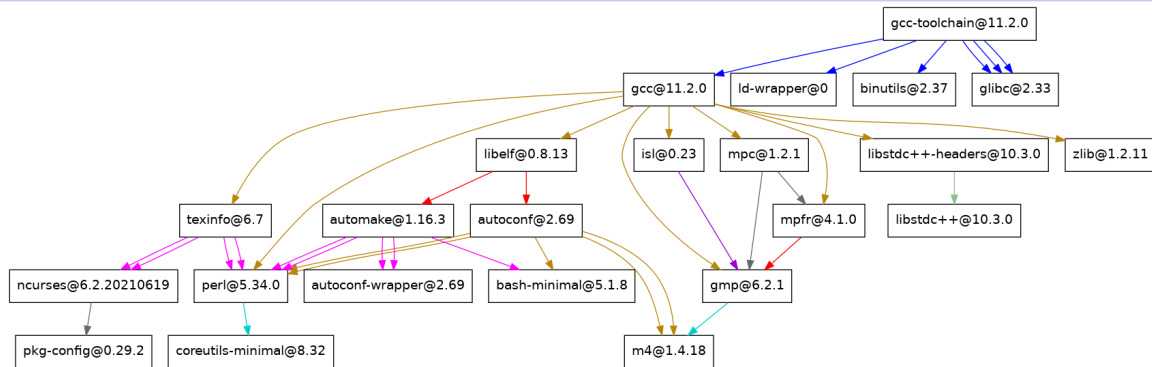
- | | |
|---|---------------------------------|
| ① How to reinstall this exact same set of packages? | <code>apt-get update</code> |
| ② How to inspect the base image? | <code>FROM debian:stable</code> |

What if answering one question answers the other?

How to capture?

When Alice says « GCC at version 11.2.0 »

guix graph

Is it the same “version” of GCC if `mpfr` is replaced by version 4.0 ?complete graph: 43 ou 104 ou 125 ou 218 nodes
(depending what we consider as *binary seed* for *bootstrapping*)

How to capture?

What does reproducing a computational environment mean?

Alice says "GCC at version 11.2.0"

All the tools (node of the graph) must be captured!

Remember

```
#include <stdio.h>
#include <math.h>
```

```
int main(){
    printf("%E\n", j0f(0x1.33d152p+1f));
}
```

```
alice@laptop$ gcc bessell.c                && ./a.out
```

```
5.643440E-08
```

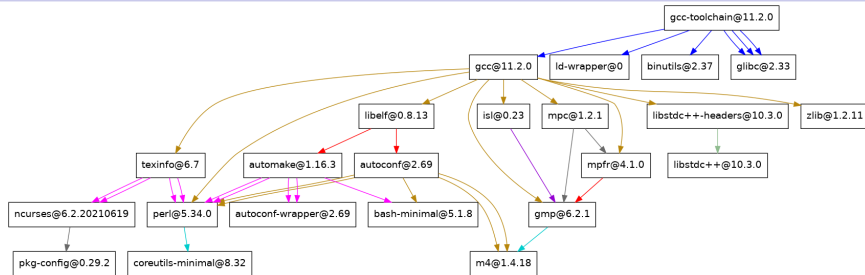
```
carole@desktop$ gcc bessell.c -lm -fno-builtin && ./a.out
```

```
5.963430E-08
```

(due to *constant folding*)

State = Directed Acyclic Graph(DAG)

package manager = graph manager



Each node specifies a recipe defining:

- ▶ code source
- ▶ build-time tools
- ▶ dependencies

and potentially some *ad-hoc* modifications (patch)
compilers, build automation, configuration flags etc.
other packages (→recursive ~> graph)

Complete graph : Python = 137 nodes, Numpy = 189, Matplotlib = 915, Scipy = 1439 nodes

Revision = one specific graph

« GCC at version 11.2.0 » = one pinned graph

```
$ guix describe
```

```
Generation 76 Apr 25 2022 12:44:37 (current)
```

```
guix eb34ff1
```

```
repository URL: https://git.savannah.gnu.org/git/guix.git
```

```
branch: master
```

```
commit: eb34ff16cc9038880e87e1a58a93331fca37ad92
```

this revision eb34ff1 captures the **complete** graph

- ▶ Alice says « I used Guix at revision eb34ff1 »
- ▶ Blake knows all for reproducing the same environment

Collaboration in action

Guix is helping

Alice

describes her environment:

- ▶ the list of the tools using the file `manifest.scm`, spawns her environment e.g.,
`guix shell -m manifest.scm`

Collaboration in action

Guix is helping

Alice

describes her environment:

- ▶ the list of the tools using the file `manifest.scm`, spawns her environment e.g.,
`guix shell -m manifest.scm`
- ▶ the revision (Guix itself and potentially all the other channels)
`guix describe -f channels > state-alice.scm`

Collaboration in action

Guix is helping

collaborate = share one computational environment

Alice

describes her environment:

- ▶ the list of the tools using the file `manifest.scm`, spawns her environment e.g.,
`guix shell -m manifest.scm`
- ▶ the revision (Guix itself and potentially all the other channels)
`guix describe -f channels > state-alice.scm`

Collaboration in action

Guix is helping

collaborate = share one computational environment \Rightarrow share one specific graph

Alice

describes her environment:

- ▶ the list of the tools using the file `manifest.scm`, spawns her environment e.g.,
`guix shell -m manifest.scm`
- ▶ the revision (Guix itself and potentially all the other channels)
`guix describe -f channels > state-alice.scm`

Collaboration in action

Guix is helping

collaborate = share one computational environment \Rightarrow share one specific graph

Alice

describes her environment:

- ▶ the list of the tools using the file `manifest.scm`, spawns her environment e.g.,
`guix shell -m manifest.scm`
- ▶ the revision (Guix itself and potentially all the other channels)
`guix describe -f channels > state-alice.scm`
- ▶ then **shares these two files**: `state-alice.scm` and `manifest.scm`

Collaboration in action

Guix is helping

collaborate = share one computational environment \Rightarrow share one specific graph

Alice

describes her environment:

- ▶ the list of the tools using the file `manifest.scm`, spawns her environment e.g.,
`guix shell -m manifest.scm`
- ▶ the revision (Guix itself and potentially all the other channels)
`guix describe -f channels > state-alice.scm`
- ▶ then **shares these two files**: `state-alice.scm` and `manifest.scm`

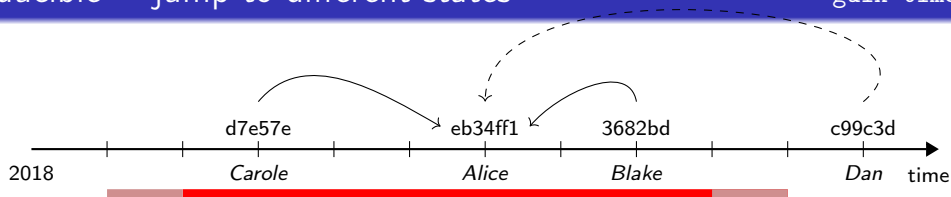
Blake

spawns the same computational environment **from these two files**

```
guix time-machine -C state-alice.scm -- shell -m manifest.scm
```


Reproducible = jump to different states

guix time-machine



Requirements for being reproducible with the passing of time using Guix:

- ▶ Preservation of the **all** source code
- ▶ *Backward* compatibility of the Linux kernel
- ▶ Compatibility of *hardware* (to some extent)
- ▶ (No time-bomb!)

What is the size of this temporal window where these 3 conditions are satisfied?

To my knowledge, the Guix project is quasi-unique by experimenting since v1.0 in 2019.

how to redeploy later and elsewhere what has been deployed today and here?

Traceability and transparency

being collectively able to study bug-to-bug

Guix should manage everything

about the **environment**

```
guix time-machine -C state.scm -- shell -m list-software.scm
```

if it is specified

« how to build »

channels.scm (state)

« what to build »

manifest.scm (software list)

how to redeploy later and elsewhere what has been deployed today and here?

Traceability and transparency

being collectively able to study bug-to-bug

Guix should manage everything

about the **environment**

```
guix time-machine -C state.scm -- shell -m list-software.scm
```

if it is specified

« how to build »

channels.scm (state)

« what to build »

manifest.scm (software list)

auto-promo:

Toward practical transparent verifiable and long-term reproducible research using Guix [\(link\)](#)

how to redeploy later and elsewhere what has been deployed today and here?

Traceability and transparency

being collectively able to study bug-to-bug

Guix should manage everything

about the **environment**

```
guix time-machine -C state.scm -- shell -m list-software.scm
```

if it is specified

« how to build »

channels.scm (state)

« what to build »

manifest.scm (software list)

auto-promo:

Toward practical transparent verifiable and long-term reproducible research using Guix [\(link\)](#)

What is required in addition to these 2 files?

Still issues!

Guix cannot fix all the broken world, isn't it?

(opinionated)



Still issues!

Guix cannot fix all the broken world, isn't it?

(opinionated)



► Which one is efficient?

Still issues!

Guix cannot fix all the broken world, isn't it?

(opinionated)



- ▶ Which one is efficient? It depends on efficient... fast? torque? weight?

Still issues!

Guix cannot fix all the broken world, isn't it?

(opinionated)



- ▶ Which one is efficient? It depends on efficient... fast? torque? weight?
- ▶ Which one is robust?

Still issues!

Guix cannot fix all the broken world, isn't it?

(opinionated)



- ▶ Which one is efficient? It depends on efficient... fast? torque? weight?
- ▶ Which one is robust? I know which one I choose for going through the unknown.

Still issues!

Guix cannot fix all the broken world, isn't it?

(*opinionated*)



- ▶ Which one is efficient? It depends on efficient... fast? torque? weight?
- ▶ Which one is robust? I know which one I choose for going through the unknown.

Easy vs Complicated
vs
Simple vs Complex

Easy: near to our skill, familiar (\approx relative)

Simple: one task, one concept (\approx predictable)

Still issues!

Guix cannot fix all the broken world, isn't it?

(opinionated)



- ▶ Which one is efficient? It depends on efficient... fast? torque? weight?
- ▶ Which one is robust? I know which one I choose for going through the unknown.

Easy vs Complicated
vs
Simple vs Complex

Easy: near to our skill, familiar (\approx relative)
Simple: one task, one concept (\approx predictable)

Rule of thumb

- ▶ Composing simple systems builds complex and robust systems
- ▶ Complex and easy systems are complicated thus fragile
- ▶ If you have no idea where to start for auditing a tool, it's suspicious!

Still issues!

(Un)Reproducible research

(opinionated)

the main issue is more about our collective practises

Still issues!

(Un)Reproducible research

(opinionated)

the main issue is more about our collective practises
than about technical limitations of our tools

Still issues!

(Un)Reproducible research

(*opinionated*)

the main issue is more about our collective practises

than about technical limitations of our tools

Technical Roadblocks

random pick of four among others

- ① How to audit pre-trained Machine Learning models?
- ② Hardware evolution over project duration (2-10 years)
- ③ Computational environment (deployment) *bit-for-bit* reproducible is reachable!
Bit-for-bit reproducible computation is more difficult. Does it make sense?
- ④ What is the size of the *binary* seed rooting the graph of dependencies?
language: Haskell, OCaml, Rust, etc.

Still issues!

(Un)Reproducible research

(*opinionated*)

the main issue is more about our collective practises

than about technical limitations of our tools

Technical Roadblocks

random pick of four among others

- ① How to audit pre-trained Machine Learning models?
- ② Hardware evolution over project duration (2-10 years)
- ③ Computational environment (deployment) *bit-for-bit* reproducible is reachable!
Bit-for-bit reproducible computation is more difficult. Does it make sense?
- ④ What is the size of the *binary* seed rooting the graph of dependencies?
language: Haskell, OCaml, Rust, etc.

What the time will eat is unknown

consider efficient as robust (and frugal) then the rest, eventually

ACM REP 24: Conference on Reproducibility and Replicability

- ▶ **The Impact of Hardware Variability on Applications Packaged with Docker and Guix: a Case Study in Neuroimaging** [\(link\)](#)
we study the effect of nine different CPU models using two software packaging systems (Docker and Guix), and we compare the resulting hardware variability to numerical variability measured with random rounding.
- ▶ **Embracing Deep Variability For Reproducibility and Replicability** [\(link\)](#)
we delve into the application of software engineering techniques, specifically variability management, to systematically identify and explicit points of variability that may give rise to reproducibility issues (eg language, libraries, compiler, virtual machine, OS, environment variables, etc).

Reproducible deployment

(ideally)

- ▶ Alice says the tool `r-harmony` from Guix revision `eb34ff1` (revision from 2022).
- ▶ Blake runs on a different machine or at a different point in time:

```
guix time-machine --commit=eb34ff1 -- shell r-harmony
```

and Blake deploys the *exact same software environment*, bit-for-bit.

Reproducible deployment

(ideally)

- ▶ Alice says the tool `r-harmony` from Guix revision `eb34ff1` (revision from 2022).
- ▶ Blake runs on a different machine or at a different point in time:

```
guix time-machine --commit=eb34ff1 -- shell r-harmony
```

and Blake deploys the *exact same software environment*, bit-for-bit.

Under the assumptions

- ▶ All the source code is still publicly available. (e.g., more than 477)
- ▶ All the intermediary builds are deterministic. reproducible-builds.org (link)

Reproducible deployment

(ideally)

- ▶ Alice says the tool `r-harmony` from Guix revision `eb34ff1` (revision from 2022).
- ▶ Blake runs on a different machine or at a different point in time:

```
guix time-machine --commit=eb34ff1 -- shell r-harmony
```

and Blake deploys the *exact same software environment*, bit-for-bit.

Under the assumptions

- ▶ All the source code is still publicly available. (e.g., more than 477)
- ▶ All the intermediary builds are deterministic. reproducible-builds.org (link)

is all the source code still publicly available?

Still publicly available?

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Dec. 2022
	v1.4.0
#sources	20 184
avail.	96.4%
missing	3.6%
hash mis.	52

Still publicly available?

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Dec. 2022
	v1.4.0
#sources	20 184
avail.	96.4%
missing	3.6%
hash mis.	52

- ▶ `openjdk-9.181.tar.bz2` is unavailable
from its original upstream URL as it appears in Guix v1.4.0.

Still publicly available?

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Dec. 2022
	v1.4.0
#sources	20 184
avail.	96.4%
missing	3.6%
hash mis.	52

- ▶ `openjdk-9.181.tar.bz2` is unavailable
from its original upstream URL as it appears in Guix v1.4.0.
- ▶ `openjdk@9.181` had 184 dependents
loosing it \implies loosing 185 packages, not one.

Still publicly available?

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	15 520	20 184
avail.	95.7%	96.4%
missing	4.3%	3.6%
hash mis.	66	52

- ▶ `openjdk-9.181.tar.bz2` is unavailable
from its original upstream URL as it appears in Guix v1.4.0.
- ▶ `openjdk@9.181` had 184 dependents
loosing it \implies loosing 185 packages, not one.

Still publicly available?

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Nov. 2020 v1.2.0	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	13 609	15 520	20 184
avail.	95.0%	95.7%	96.4%
missing	5.0%	4.3%	3.6%
hash mis.	69	66	52

- ▶ `openjdk-9.181.tar.bz2` is unavailable
from its original upstream URL as it appears in Guix v1.4.0.
- ▶ `openjdk@9.181` had 184 dependents
loosing it \implies loosing 185 packages, not one.

Still publicly available?

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	Apr. 2020 v1.1.0	Nov. 2020 v1.2.0	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	11 659	13 609	15 520	20 184
avail.	92.4%	95.0%	95.7%	96.4%
missing	7.6%	5.0%	4.3%	3.6%
hash mis.	63	69	66	52

- ▶ `openjdk-9.181.tar.bz2` is unavailable
from its original upstream URL as it appears in Guix v1.4.0.
- ▶ `openjdk@9.181` had 184 dependents
loosing it \implies loosing 185 packages, not one.

Still publicly available?

“Link rot” empirical evaluation = the problem

(completed mid-2024)

	May 2019 v1.0.0	Apr. 2020 v1.1.0	Nov. 2020 v1.2.0	May 2021 v1.3.0	Dec. 2022 v1.4.0
#sources	8 794	11 659	13 609	15 520	20 184
avail.	91.5%	92.4%	95.0%	95.7%	96.4%
missing	8.5%	7.6%	5.0%	4.3%	3.6%
hash mis.	87	63	69	66	52

- ▶ `openjdk-9.181.tar.bz2` is unavailable
from its original upstream URL as it appears in Guix v1.4.0.
- ▶ `openjdk@9.181` had 184 dependents
loosing it \implies loosing 185 packages, not one.

Software Heritage comes in!

Like all digital information, source code is fragile

link rot: projects are created, moved around, removed

“too big to fail”: e.g., Gitorious.org, Google Code, Bitbucket, etc.

Software Heritage comes in!

Like all digital information, source code is fragile

link rot: projects are created, moved around, removed

“too big to fail”: e.g., Gitorious.org, Google Code, Bitbucket, etc.

If a website disappears, you go to the Internet Archive. . .

Where do you do if (a repository on) GitHub or GitLab goes away?

Software Heritage comes in!

Like all digital information, source code is fragile

link rot: projects are created, moved around, removed

“too big to fail”: e.g., Gitorious.org, Google Code, Bitbucket, etc.

If a website disappears, you go to the Internet Archive. . .

Where do you do if (a repository on) GitHub or GitLab goes away?

Answer: **Software Heritage**

Software Heritage comes in!

Like all digital information, source code is fragile

link rot: projects are created, moved around, removed

“too big to fail”: e.g., Gitorious.org, Google Code, Bitbucket, etc.



collect, preserve and share source code

If a website disappears, you go to the Internet Archive. . .

Where do you do if (a repository on) GitHub or GitLab goes away?

Answer: **Software Heritage**

The SWH archive is the largest publicly available archive of software source code.

An universal source code archive

Software Heritage: international + non-profit

built with long term in mind

Sharing the vision



www.softwareheritage.org/support/testimonials

Donors, members, sponsors



Diamond sponsor



Platinum sponsors



Gold sponsors



Silver sponsors



Bronze sponsors



www.softwareheritage.org/support/sponsors

widely supported

SWH, pillar of Open Science, pillar of Reproducible Research!

Software Heritage in a nutshell: SWH

www.softwareheritage.org



Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better scientific outcome for all

SWH, pillar of Open Science, pillar of Reproducible Research!

Software Heritage in a nutshell: SWH

www.softwareheritage.org



Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better scientific outcome for all

Reference catalog



find and reference
all software source code

SWH, pillar of Open Science, pillar of Reproducible Research!

Software Heritage in a nutshell: SWH

www.softwareheritage.org



Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better scientific outcome for all

Reference catalog



find and reference
all software source code

Universal archive



preserve and share
all software source code

SWH, pillar of Open Science, pillar of Reproducible Research!

Software Heritage in a nutshell: SWH

www.softwareheritage.org



Collect, preserve and share *all* software source code

Preserving our heritage, enabling better software and better scientific outcome for all

Reference catalog



find and reference
all software source code

Universal archive



preserve and share
all software source code

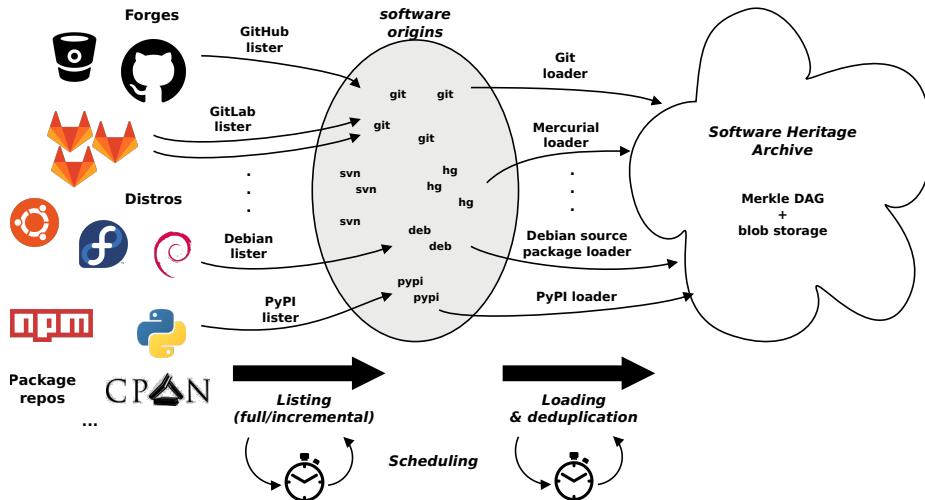
Research infrastructure



enable analysis
of all software source code

Ingest well-known public source code warehouses

Data flow: Harvest as much as publicly available!



Ingest well-known public source code warehouses

SWH archive coverage

archive.softwareheritage.org



Note: the counters and graphs are based on heuristics that might not reflect the exact size of the archive. While the long-term trends shown and ballpark figures are reliable, individual point-in-time values might not be.



Ingest well-known public source code warehouses

SWH archive coverage

archive.softwareheritage.org



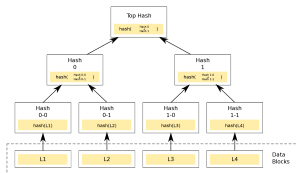
Note: the counters and graphs are based on heuristics that might not reflect the exact size of the archive. While the long-term trends shown and ballpark figures are reliable, individual point-in-time values might not be.



- ▶ on disk: ~1 PiB; as a graph ~35 B nodes, ~500 B edges
- ▶ the largest public source code archive in the world (and growing!)

SWH archive: a (giant) Merkle DAG

Merkle tree (R. C. Merkle, CRYPTO 1987)

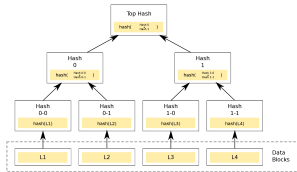


Combination of tree and hash function

Classical cryptographic construction

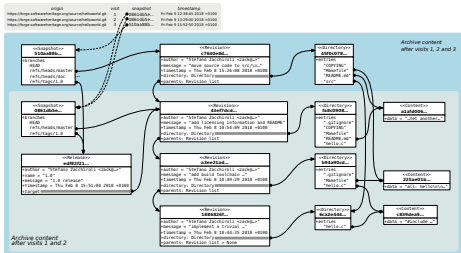
- ▶ fast, parallel signature of large data structures
- ▶ widely used (e.g., Git, blockchains, IPFS, ...)
- ▶ built-in deduplication

Merkle tree (R. C. Merkle, CRYPTO 1987)



Classical cryptographic construction

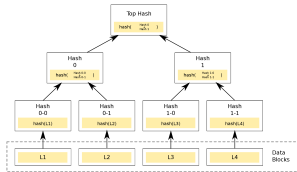
- ▶ fast, parallel signature of large data structures
- ▶ widely used (e.g., Git, blockchains, IPFS, ...)
- ▶ built-in deduplication



- ▶ A **global graph** linking together
- ▶ fully **deduplicated** source code artifact
(files, commits, directories, releases, etc.)
- ▶ providing a **unified view**
on the entire *Software Commons*.

SWH archive: a (giant) Merkle DAG

Merkle tree (R. C. Merkle, CRYPTO 1987)

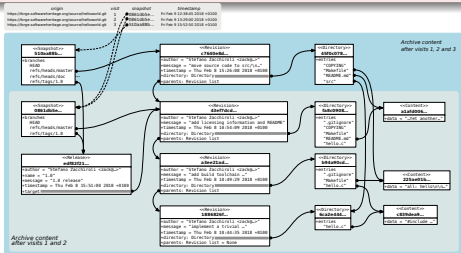


Combination of tree and hash function

Classical cryptographic construction

- ▶ fast, parallel signature of large data structures
- ▶ widely used (e.g., Git, blockchains, IPFS, ...)
- ▶ built-in deduplication

find and reference: SWHID

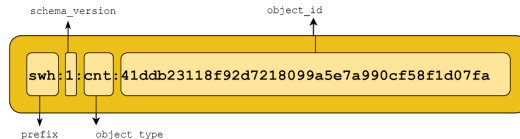


- ▶ A **global graph** linking together
- ▶ fully **deduplicated** source code artifact
(files, commits, directories, releases, etc.)
- ▶ providing a **unified view**
on the entire *Software Commons*.

Content-addressed reference, intrinsic identifier, inherent identifier

Software Heritage Identifiers (SWHIDs)

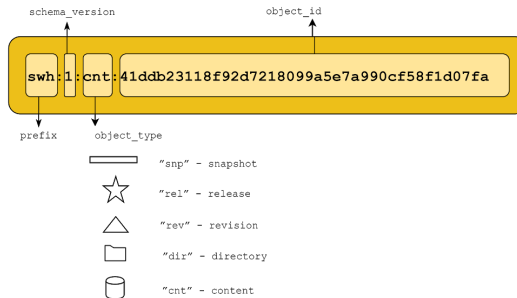
swhid.org



Content-addressed reference, intrinsic identifier, inherent identifier

Software Heritage Identifiers (SWHIDs)

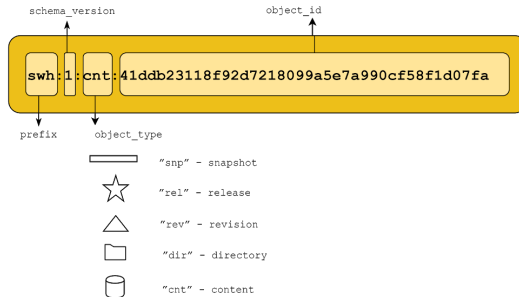
swhid.org



Content-addressed reference, intrinsic identifier, inherent identifier

Software Heritage Identifiers (SWHIDs)

swhid.org

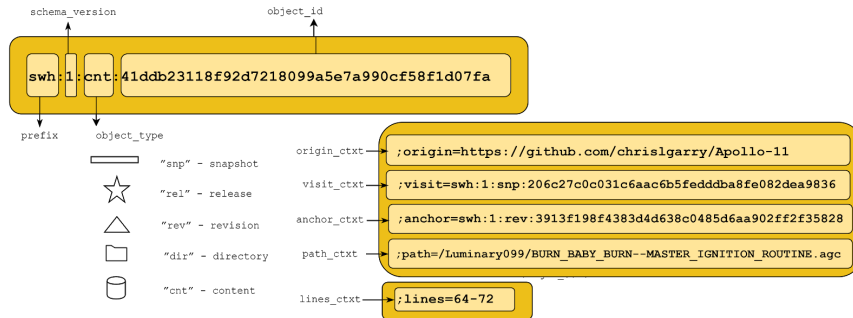


<https://archive.softwareheritage.org/swh:1:cnt:41ddb23118f92d7218099a5e7a990cf58f1d07fa>

Content-addressed reference, intrinsic identifier, inherent identifier

Software Heritage Identifiers (SWHIDs)

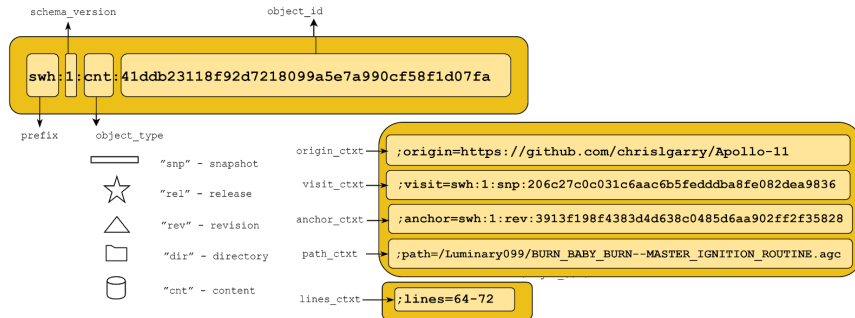
swhid.org



Content-addressed reference, intrinsic identifier, inherent identifier

Software Heritage Identifiers (SWHIDs)

swhid.org

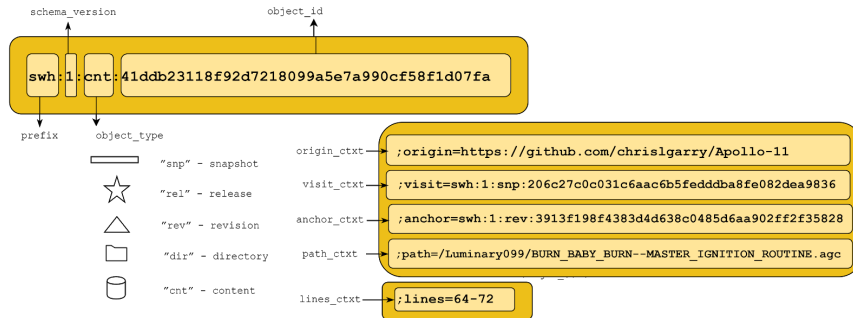


demo! (link)

Content-addressed reference, intrinsic identifier, inherent identifier

Software Heritage Identifiers (SWHIDs)

swhid.org



demo! ([link](#))

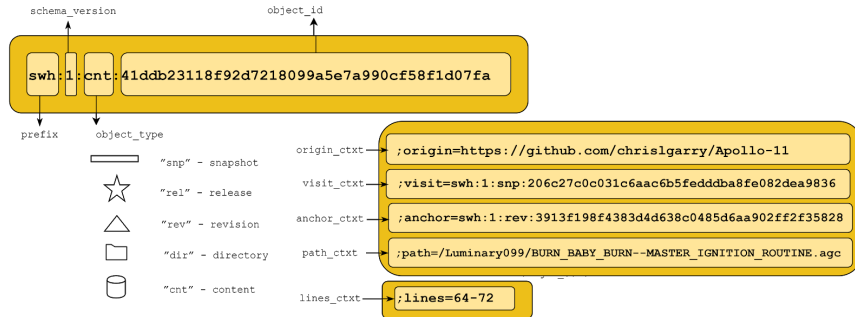
an emerging standard

HOWTO archive and reference your code ([link](#))

Content-addressed reference, intrinsic identifier, inherent identifier

Software Heritage Identifiers (SWHIDs)

swhid.org



[demo!](#) (link)

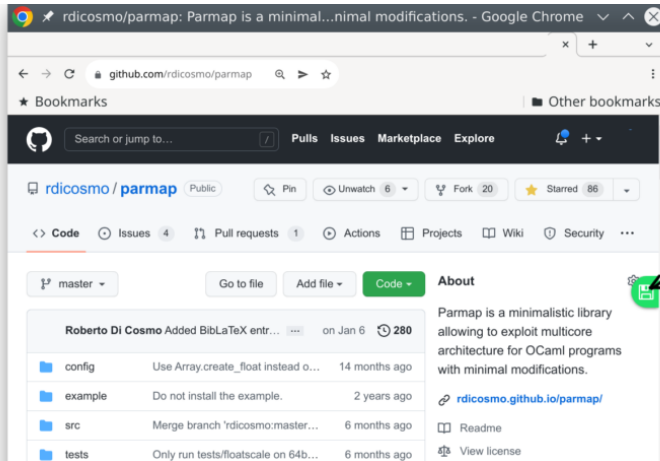
[demo save!](#) (link)

an emerging standard

[HOWTO archive and reference your code](#) (link)

Browser extension

<https://www.softwareheritage.org/browser-extensions>





This tab shows the archival status of the repository

- Green** up to date
- Yellow** not up to date
- Grey** not archived yet
- Red** not archivable (private)

Deposit source code via HAL

<https://doc.hal.science/deposer/deposer-le-code-source>

 **HAL** Déposer le code source d'un logiciel

  Rechercher

Accueil Guide utilisateur FAQ Questions Juridiques Tutoriels UsuHAL Les Essentiels de HAL API TripleStore Support HAL

Guide utilisateur

Déposer le code source d'un logiciel

- Quel logiciel déposer dans HAL/Software Heritage ?
- Qu'appelle-t-on un logiciel de recherche ? Quels types de logiciels sont éligibles ?
- Quels fichiers dépose-t-on ?
- 2 modalités de dépôt selon les modalités de développement
 - Option 1 : Le logiciel a été développé en dehors d'une plateforme de développement en ligne, en local

Utiliser l'identifiant pérenne SoftWare Hash IDentifier (SWHID) <

Pour reproduire une expérience, il est indispensable de connaître avec exactitude la version du logiciel, ce que permet le SoftWare Hash Identifier (SWHID). Avec le SWHID vous pouvez retrouver dans l'archive de Software Heritage vos codes, lire en ligne tous les contenus textuels et télécharger le code source.

Grâce au SWHID, l'identification des logiciels s'effectue sans passer par un registre. en effet, le SWHID est calculé à partir des données du code source lui-même, à la différence d'un identifiant de type DOI qui s'appuie sur un registre externe pour faire le lien entre un objet et sa description. Le [SWHID](#) est l'empreinte digitale du logiciel et ne dépend pas d'un résolveur: un utilisateur peut ainsi le calculer localement.

Chaque version du schéma d'identification est entretenue même quand celui-ci sera étiqueté obsolète, dans le cas de collisions sur les hachages SHA1.

Bon à savoir : si pour le dépôt dans HAL, le SWHID de type "directory" est requis, une fois que le code source est archivé dans Software Heritage, il est possible d'obtenir des SWHID pour différents artefacts : [un fichier](#), [un commit](#), quelques lignes de code source dans un fichier donné, etc. Le pré-requis est d'accepter le transfert vers Software Heritage si le dépôt a été effectué selon les modalités de l'option 1.

[En savoir plus sur le SWHID](#)

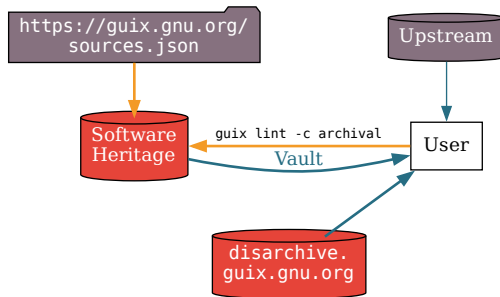
Cite software with the BibT_EX style

<https://www.softwareheritage.org/2020/05/26/citing-software-with-style>



How to rescue source code?

Guix + SWH = ♥



content-address

Guix: “normalized archive”

nar + sha256

SWH: SWHID

Git compatible sha1

Guix the first free software distribution and tool backed by the stable SWH archive

ACM REP 24: Conference on Reproducibility and Replicability

► Source Code Archiving to the Rescue of Reproducible Deployment [\(link\)](#)

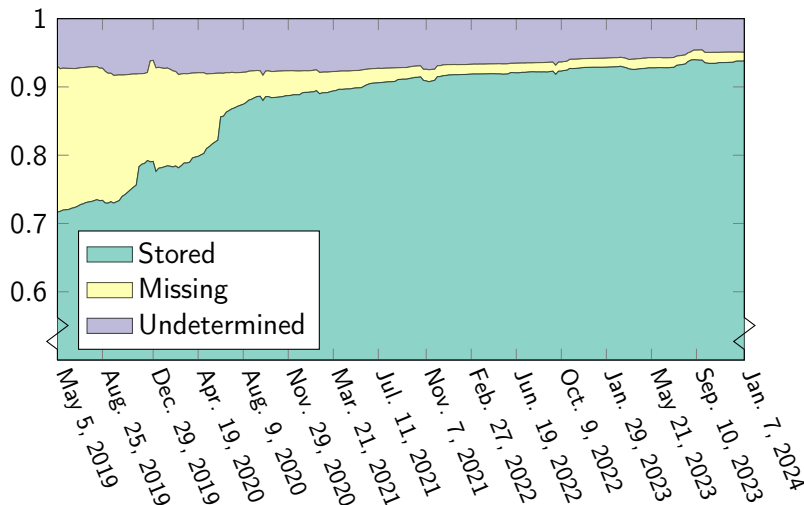
We describe our work connecting Guix with Software Heritage, the universal source code archive, making Guix the first free software distribution and tool backed by a stable archive.

► Longevity of Artifacts in Leading Parallel and Distributed Systems Conferences: a Review of the State of the Practice in 2023 [\(link\)](#)

By reviewing the methods and tools used to create and share artifacts in a technical, in-depth, and article content-agnostic manner, we found that the state of practice does not address reproducibility in terms of artifact longevity and we expose eight observations that support this finding.

How to rescue source code?

Coverage by sampled Guix revision



The end?

Concretely, does it work for real?

```
guix time-machine --commit=6298c3ffd96 -- install r-harmony
```

Installs (and potentially rebuilds) Harmony defined in Guix 1.0.0 from 2019.

The end?

Concretely, does it work for real?

```
guix time-machine --commit=6298c3ffd96 -- install r-harmony
```

Installs (and potentially rebuilds) Harmony defined in Guix 1.0.0 from 2019.

- ▶ This command exploits SWH support as it was in 2019: in its infancy.
- ▶ Recovery mechanism is itself improving over time.

The end?

Concretely, does it work for real?

```
guix time-machine --commit=6298c3ffd96 -- install r-harmony
```

Installs (and potentially rebuilds) Harmony defined in Guix 1.0.0 from 2019.

- ▶ This command exploits SWH support as it was in 2019: in its infancy.
- ▶ Recovery mechanism is itself improving over time.

Rebuilding the whole only from SWH

- ▶ June 2023 redoing paper from 2020 ([link](#))
- ▶ December 2023 redoing paper from 2022 ([link](#))

The end?

Concretely, does it work for real?

```
guix time-machine --commit=6298c3ffd96 -- install r-harmony
```

Installs (and potentially rebuilds) Harmony defined in Guix 1.0.0 from 2019.

- ▶ This command exploits SWH support as it was in 2019: in its infancy.
- ▶ Recovery mechanism is itself improving over time.

Rebuilding the whole only from SWH

- ▶ June 2023 redoing paper from 2020 [\(link\)](#)
- ▶ December 2023 redoing paper from 2022 [\(link\)](#)

Two main difficulties remain:

- ▶ Bootstrapping
 - binary seed rooting the graph of dependencies
 - storing the seed itself and rebuilding from the seed, if needed
- ▶ Time bomb
 - deterministic build depends on date

we can fix the future not the past

Adventures on the quest for long-term reproducible deployment [\(link\)](#)

So, convinced?

“few” and “basic” needs for software

reminder, 4 letters: ARDC

Archive

Research software artifacts must be properly **archived**make sure we can *retrieve* them (*reproducibility*)

Reference

Research software artifacts must be properly **referenced**make sure we can *identify* them (*reproducibility*)

Describe

Research software artifacts must be properly **described**make it easy to *discover* and *reuse* them (*visibility*)

Cite/Credit

Research software artifacts must be properly **cited** (*not the same as referenced!*)to give *credit* to authors (*evaluation!*)

So, convinced?

Reproducible computational environment, when?

(opinionated)

when collective practises will stop to promote *engineering methods*

engineering method

what do we **gain** compared to current?

vs

science method

what do we **understand** compared to current?

efficient vs robust

user autonomy¹

Thanks Guix and Software Heritage, the situation is improving over the years

¹digital sovereignty



Mnemosyne
pearls
memory



Cronus
scythe
time

We cannot predict beforehand
what the scythe will cut

Pearls

- ▶ simple made easy
- ▶ efficient means robust
- ▶ content-addressed, intrinsic identifier, inherent reference
- ▶ transparent and auditable computational environment
- ▶ focus on user-autonomy

Are Guix and Software Heritage two pearls against the scythe?

Let make better scientific outcomes!

it depends on our collective practices

use, adopt, advocate



<https://hpc.guix.info>



<https://softwareheritage.org>

contribute, fund, support, join

Join the fun!

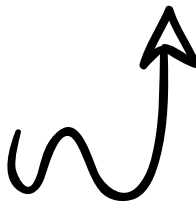
The vision to *reach*



Software Heritage



Guix



The Re**Science** Journal

Questions?



<https://www.softwareheritage.org>



<https://hpc.guix.info/events/2024-2025/café-guix>



<https://www.recherche-reproductible.fr>

Appendix

Some papers

enable research

- ▶ *Source Code Archiving to the Rescue of Reproducible Deployment*
Ludovic Courtès, Timothy Sample, Simon Tournier, Stefano Zacchiroli
ACM REP '24: Proceedings of the 2nd ACM Conference on Reproducibility and Replicability.
- ▶ *A Large-scale Dataset of (Open Source) License Text Variants*
Stefano Zacchiroli
MSR 2022 (best dataset paper) + Empir. Soft. Eng. 28(6): 147 (2023)
- ▶ *Geographic diversity in public code contributions*
Davide Rossi, Stefano Zacchiroli
MSR 2022
- ▶ *Gender differences in public code contributions: a 50-year perspective*
Stefano Zacchiroli
IEEE Software, 2021

Why preserving?

Because **online services sometimes stop**

- ▶ Google Code (link) early 2016
- ▶ Alioth (Debian) in 2018 replaced by Salsa
- ▶ Gna! in 2017 after 13 years
- ▶ Gitourious in 2015 (the second most popular service for hosting Git repository in 2011)
- ▶ etc.

Why preserving?

Because **online services sometimes stop**

- ▶ Google Code (link) early 2016
- ▶ Alioth (Debian) in 2018 replaced by Salsa
- ▶ Gna! in 2017 after 13 years
- ▶ Gitourious in 2015 (the second most popular service for hosting Git repository in 2011)
- ▶ etc.
- ▶ `gforge.inria.fr` for example Guix issue #42162 (link)

Believe it or not, `gforge.inria.fr` was finally phased out on Sept. 30th. And believe it or not, despite all the work and all the chat :-), we lost the source tarball of Scotch 6.1.1 for a short period of time (I found a copy and uploaded it to berlin a couple of hours ago).

How to preserve?

Forge \neq Archive

collaborative software platform for developing

L'objectif d'une forge est de permettre à plusieurs développeurs de **participer ensemble au développement** d'un ou plusieurs logiciels, le plus souvent à travers le réseau Internet.

[https://fr.wikipedia.org/wiki/Forge_\(informatique\)](https://fr.wikipedia.org/wiki/Forge_(informatique))

(no English wikipedia entry)

L'archivage est un ensemble d'actions qui a pour but de garantir l'accessibilité sur le long terme d'informations (dossiers, documents, données) que l'on doit ou souhaite conserver pour des raisons juridiques

<https://fr.wikipedia.org/wiki/Archivage>

Software Heritage « *are building the universal software archive* » (link)

Recipe for defining a package

one node of the graph

```
(define python ;definition of the node python
  (package
    (name "python")
    (version "3.9.9")
    (source ... ) ;points to URI source code
    (build-system gnu-build-system) ;./configure & make
    (arguments ... ) ; configure flags, etc.
    (inputs (list bzip2 ;other nodes -> graph (DAG)
                  expat gdbm libffi sqlite ...))))
```

- ▶ Each inputs is similarly defined (recursion → graph)
- ▶ There is no cycle (bzip2 or its inputs cannot refer to python)

What are the roots of the graph? Part of the broad *bootstrapping* (link) problem

```
(define-public python-scikit-learn
  (package
    (name "python-scikit-learn")
    (version "1.4.2")
    (source
      (origin
        (method git-fetch)
        (uri (git-reference
              (url "https://github.com/scikit-learn/scikit-learn")
              (commit version)))
        (sha256
          (base32
            "0pdd508c9540x9qimq83b8kspb6mb98w7w7i7lnb1jqj7rijal6f")))
        ;; various fields omitted
```

Source code identification

origin specifies:

method: *tarball*, VCS as Git, Mercurial, Subversion, etc.

uri: upstream location (URL)

sha256: cryptographic hash

Source code identification

origin specifies:

method: *tarball*, VCS as Git, Mercurial, Subversion, etc.

uri: upstream location (URL)

sha256: cryptographic hash

source code is *essentially* content-addressed

Source code identification

origin specifies:

method: *tarball*, VCS as Git, Mercurial, Subversion, etc.

uri: upstream location (URL)

sha256: cryptographic hash

source code is *essentially* **content-addressed**

If **uri** becomes stale, e.g., URL no longer available or tempered (hash mismatch)

Source code identification

origin specifies:

method: *tarball*, VCS as Git, Mercurial, Subversion, etc.

uri: upstream location (URL)

sha256: cryptographic hash

source code is *essentially* **content-addressed**

If uri becomes stale, e.g., URL no longer available or tempered (hash mismatch)
Then Blake can work around. (if a copy is available elsewhere)

Source code identification

origin specifies:

method: *tarball*, VCS as Git, Mercurial, Subversion, etc.

uri: upstream location (URL)

sha256: cryptographic hash

source code is *essentially* content-addressed

If uri becomes stale, e.g., URL no longer available or tempered (hash mismatch)
Then Blake can work around. (if a copy is available elsewhere)

Elsewhere might be

- ▶ a new URL

guix download finds the source with the expected hash and proceeds.

Source code identification

origin specifies:

method: *tarball*, VCS as Git, Mercurial, Subversion, etc.

uri: upstream location (URL)

sha256: cryptographic hash

source code is *essentially* **content-addressed**

If uri becomes stale, e.g., URL no longer available or tempered (hash mismatch)
Then Blake can work around. (if a copy is available elsewhere)

Elsewhere might be

- ▶ a new URL

guix download finds the source with the expected hash and proceeds.

- ▶ a **content-addressed server**

as served by the Guix project or the Nix project, or the Software Heritage initiative.

Fallback in action

```
$ guix time-machine -C channels.scm -- shell -m manifest.scm
Updating channel 'guix' from Git repository at 'https://git.savannah.gnu.org/gi
Updating channel 'example' from Git repository at 'https://whatever-here.org/do
SWH: found revision 67c9f2143aa6f545419ae913b4ae02af4cd3effc with directory at
SWH vault: requested bundle cooking, waiting for completion...
swh:1:rev:67c9f2143aa6f545419ae913b4ae02af4cd3effc.git/
[...]
fatal: could not read Username for 'https://github.com': No such device or addr
Trying content-addressed mirror at berlin.guix.gnu.org...
Trying to download from Software Heritage...
SWH: found revision eleefd033b8a2c4c81bab6fde08ebb116c6abb8 with directory at'
[...]
```

<https://simon.tournier.info/posts/2021-10-25-software-heritage.html>

CC-BY 4.0

<https://creativecommons.org/licenses/by/4.0>

Images from Gmsh and GetDP projects, slide: 1.

Images from Wikipedia, slide: 1.

Courtesy of Software Heritage team, slides: 8, 10, 29, 30, 32, 33, 34, 35, 41.